**Problem statement 1. Consider following Bank database schema and solve given queries:**

**Account(Acc\_no, branch\_name,balance) branch(branch\_name,branch\_city, assets) customer(cust\_name,cust\_street,cust\_city) Depositor(cust\_name,acc\_no)**

**Loan(loan\_no,branch\_name,amount) Borrower(cust\_name,loan\_no)**

**Q.1 Create above tables with appropriate constraints like primary key, foreign key, not null etc.**

**with suitable data**

SQL> create table branch(branch\_name char(20) primary key,branch\_city char(20) , assets number);

Table created.

SQL> insert into branch values('Nigdi','Pune',10000000)

SQL> insert into branch values('Akurdi','Pune',50000000); SQL> insert into branch values('Khed','Pune',70000000);

SQL> insert into branch values('Chinchwad','Pune',80000000); SQL> insert into branch values('Wafgaon','Khed',30000000);

SQL> select \* from branch;

# BRANCH\_NAME BRANCH\_CITY ASSETS

|  |  |  |
| --- | --- | --- |
| Nigdi | Pune | 10000000 |
| Akurdi | Pune | 50000000 |

Khed Pune 70000000

Chinchwad Pune 80000000

Wafgaon Khed 30000000

SQL> create table account(acc\_no number primary key , branch\_name char(20) references branch(branch\_name) on delete set null ,balance number);

Table created.

SQL> insert into account values(1234567891,'Nigdi',10000); SQL> insert into account values(1234567892,'Nigdi',20000); SQL> insert into account values(1234567893,'Khed',30000);

SQL> insert into account values(1234567894,'Chinchwad',40000); SQL> select \* from account;

# ACC\_NO BRANCH\_NAME BALANCE

1234567891 Nigdi 10000

1234567892 Nigdi 20000

1234567893 Khed 30000

1234567894 Chinchwad 40000 SQL> create table customer(cust\_name char(20) primary key,cust\_street varchar(30),cust\_city char(20) );

Table created.

SQL> insert into customer values(‘Harshad’,'khed','Pune'); SQL> insert into customer values('Aviraj','Nigdi11','Pune'); SQL> insert into customer values('Nayan','Nigdi12','Pune');

SQL> insert into customer values('Ashirwad','Nigdi13','Pandharpur'); SQL> insert into customer values('Aditesh','audh','Pune');

SQL> insert into customer values('Vedant','Khed','Pune'); SQL> insert into customer values('Aditya','Khed','Pune');

SQL> insert into customer values('Siddhesh','Kolhapur','Kolhapur'); SQL> insert into customer values('Rushi','Kolhapur','Kolhapur');

SQL> insert into customer values('Sarang','Satara','Satara'); SQL> select \* from customer;

# CUST\_NAME CUST\_STREET CUST\_CITY

|  |  |  |
| --- | --- | --- |
| Harshad | Khed | Pune |
| Aviraj | Nigdi11 | Pune |
| Nayan | Nigdi12 | Pune |
| Ashirwad | Nigdi13 | Pandharpur |
| Aditesh | audh | Pune |
| Vedant | Khed | Pune |
| Aditya | Khed | Pune |
| Siddhesh | Kolhapur | Kolhapur |
| Rushi | Kolhapur | Kolhapur |
| Sarang | Satara | Satara |

10 rows selected.

SQL> create table depositor(cust\_name char(20) references customer(cust\_name) on delete cascade , acc\_no number references account(acc\_no) on delete set null);

Table created.

SQL> select \* from depositor;

# CUST\_NAME ACC\_NO

Harshad 1234567891

Aviraj 1234567892

Nayan 1234567893

Ashirwad 1234567894

SQL> create table loan(loan\_no number primary key,branch\_name char(20) references branch(branch\_name) on delete set null , amount number);

Table created.

SQL> select \* from loan;

# LOAN\_NO BRANCH\_NAME AMOUNT

|  |  |
| --- | --- |
| 9874563211 Nigdi | 200000 |
| 9874563212 Nigdi | 300000 |
| 9874563213 Nigdi | 400000 |
| 9874563214 Khed | 350000 |
| 9874563215 Chinchwad | 250000 |

SQL> create table borrower(customer\_name char(20) references customer(cust\_name)on delete cascade,loan\_no number references loan(loan\_no) on delete set null);

Table created.

SQL> select \* from borrower;

# CUSTOMER\_NAME LOAN\_NO

Aditesh 9874563211

Vedant 9874563212

Aditya 9874563213

Rushi 9874563214

**Q.2. Create synonym for customer table as cust.**

SQL> create synonym cust for customer;

Synonym created.

**Q.3 Add customer phone number in Customer table.**

SQL> alter table customer add phone\_no number(10);

Table altered.

**Q.4 Delete phone number attribute from Customer table.**

SQL> alter table customer drop column phone\_no;

Table altered.

SQL> desc customer;

Name Null? Type

CUST\_NAME NOT NULL CHAR(20)

CUST\_STREET VARCHAR2(30)

CUST\_CITY CHAR(20)

* 1. **Find the names of all branches in loan relation.**

SQL> select distinct branch\_name from loan;

# BRANCH\_NAME

Nigdi Khed

Chinchwad

* 1. **Find all customers who have a loan from bank. Find their names,loan\_no and loan amount.**

SQL> select borrower.customer\_name , borrower.loan\_no , loan.amount from borrower,loan where borrower.loan\_no = loan.loan\_no;

CUSTOMER\_NAME LOAN\_NO AMOUNT

|  |  |  |
| --- | --- | --- |
| Aditesh | 9874563211 | 200000 |
| Vedant | 9874563212 | 300000 |
| Aditya | 9874563213 | 400000 |
| Rushi | 9874563214 | 350000 |

* 1. **List all customers in alphabetical order who have loan from Nigdi branch.**

SQL> select customer\_name from borrower where loan\_no in (select loan\_no from loan where branch\_name = 'Nigdi') order by customer\_name;

# CUSTOMER\_NAME

Aditesh Aditya Vedant

* 1. **Find all customers who have an account or loan or both at bank.**

SQL> SELECT cust\_name FROM customer where cust\_name IN(SELECT customer\_name FROM borrower INTERSECT SELECT cust\_name FROM depositor);

# CUST\_NAME

Rushi

* 1. **Find average account balance at Nigdi branch.**

SQL> SELECT AVG(balance) AS avg\_balance FROM account WHERE branch\_name = 'Nigdi';

# AVG\_BALANCE

15000

* 1. **Find no. of depositors at each branch.**

SELECT branch\_name, COUNT(\*) AS num\_depositors FROM depositor GROUP BY branch\_name;

* 1. **Delete all tuples at every branch located in Nigdi.** DELETE FROM branch WHERE branch\_city = 'Nigdi'; DELETE FROM account WHERE branch\_name = 'Nigdi';

**Problem statement 2.**

1. **Consider following database schema and solve given queries**

**cust\_mstr(cust\_no,fname,lname)**

**add\_dets(code\_no,add1,add2,state,city,pincode)**

1. **Create above Tables with suitable data**
2. **Retrieve the address of customer Fname as 'xyz' and Lname as 'pqr'**
3. **Create View on add\_dets table by selecting any two columns and perform insert update delete operations**

SQL> CREATE TABLE cust\_mstr (

* 1. cust\_no INT PRIMARY KEY,
  2. fname VARCHAR(50),
  3. lname VARCHAR(50)

5 );

Table created.

SQL> select \* from cust\_mstr;

# CUST\_NO FNAME LNAME

* + 1. Harshad Karale
    2. Rushikesh Magadum
    3. Sarang Kadam

SQL> CREATE TABLE add\_dets (code\_no INT references cust\_mstr(cust\_no),add1 VARCHAR(100),add2 VARCHAR(100),state VARCHAR(50),city VARCHAR(50),pincode VARCHAR(10));

Table created.

SQL> INSERT INTO add\_dets values(1,'Jaulke BK','Pargaon','Maharashtra','Khed',410512);

1 row created.

SQL> INSERT INTO add\_dets values(2,'xyz','Kolhapur','Maharashtra','Kolhapur',410512);

1 row created

SQL> select \* from add\_dets where code\_no IN ( select cust\_no from cust\_mstr where fname='Harshad' AND lname='Karale');

1

Jaulke BK Pargaon

Maharashtra Khed 410512

1. **Create following Tables**

**emp\_mstr(e\_mpno,f\_name,l\_name,m\_name,dept,desg,branch\_no) branch\_mstr(name,b\_no)**

**List the employee details along with branch names to which they belong**

SQL> CREATE TABLE branch\_mstr(name char(10),b\_no number primary key);

Table created.

SQL> select \* from branch\_mstr; NAME B\_NO

Computer 1

# MECH 2

IT 3

Civil 4

SQL> CREATE TABLE emp\_mstr(emp\_no number NOT NULL,fname char(10),lname char(10),mname char(10),dept char(10),desg char(10),branch\_no number references branch\_mstr(b\_no) on delete set null);

Table created.

SQL> select \* from emp\_mstr;

# EMP\_NO FNAME LNAME MNAME DEPT DESG BRANCH\_NO

1 Harshad Karale Sanjay Comp Student 1

SQL> select \* from emp\_mstr where branch\_no IN(select b\_no from branch\_mstr where name = 'Computer');

# EMP\_NO FNAME LNAME MNAME DEPT DESG BRANCH\_NO

1 Harshad Karale Sanjay Comp Student 1

**Problem statement 3. Consider following Bank database schema and solve given queries:**

**Account(Acc\_no, branch\_name,balance) branch(branch\_name,branch\_city, assets) customer(cust\_name,cust\_street,cust\_city) Depositor(cust\_name,acc\_no)**

**Loan(loan\_no,branch\_name,amount) Borrower(cust\_name,loan\_no)**

**Q.1 Create above tables with appropriate constraints like primary key, foreign key constrains, not**

**null etc. with suitable data**

* 1. **Modify “assets” attribute of branch table to “Property”**

SQL> ALTER TABLE branch RENAME COLUMN assets TO property;

Table altered.

* 1. **Find all loan numbers for loans made at Akurdi Branch with loan amount > 12000.**

SQL> SELECT loan\_no FROM loan WHERE branch\_name = 'Nigdi' AND amount > 12000;

# LOAN\_NO

9874563211

9874563212

9874563213

* 1. **Find all customers who have both account and loan at bank.**

SQL> SELECT cust\_name FROM customer where cust\_name IN(SELECT customer\_name FROM borrower INTERSECT SELECT cust\_name FROM depositor);

# CUST\_NAME

Rushi

* 1. **Find all customer who have account but no loan at the bank.**

SQL> SELECT cust\_name FROM customer where cust\_name IN(SELECT customer\_name FROM borrower MINUS SELECT cust\_name FROM depositor);

# CUST\_NAME

Aviraj Rushi

* 1. **Find the average account balance at each branch**

SQL> SELECT branch\_name,AVG(balance) AS avg\_balance FROM account Group by branch\_name;

BRANCH\_NAME AVG\_BALANCE

|  |  |
| --- | --- |
| Nigdi | 15000 |
| Khed | 30000 |
| Chinchwad | 40000 |

* 1. **Find the branches where average account balance > 12000.**

Q11. Find the branches where average account balance > 12000.

SQL> SELECT AVG(balance),branch\_name FROM account GROUP BY branch\_name HAVING AVG(balance)>20000;

# AVG(BALANCE) BRANCH\_NAME

30000 Khed

40000 Chinchwad

* 1. **Find number of tuples in customer relation.**

SQL> SELECT COUNT(\*) AS tuple\_no FROM customer;

# TUPLE\_NO

12

* 1. **Calculate total loan amount given by bank.**

SQL> SELECT SUM(amount) AS total\_loan FROM loan; TOTAL\_LOAN

2143800

* 1. **Delete all loans with loan amount between 1300 and 1500.**

SQL> DELETE FROM loan WHERE amount BETWEEN 25000 AND 30000;

1 row deleted.

* 1. **Create sequence roll\_seq and use in student table for roll\_no column.**

SQL> CREATE SEQUENCE roll\_no START with 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 100 CYCLE;

Sequence created.

SQL> CREATE TABLE student(rollno number , name varchar(10));

Table created.

SQL> select \* from student;

# ROLLNO NAME

1. Harshad
2. Nayan
3. Aviraj

**Problem statement 4.**

* 1. **Create following Tables with suitable data and solve following query**

**cust\_mstr(custno,fname,lname) acc\_fd\_cust\_dets(codeno,acc\_fd\_no) fd\_dets(fd\_sr\_no,amt)**

**List the customer holding fixed deposit of amount more than 5000**

SQL> CREATE TABLE fd\_dets(fd\_sr\_no number primary key ,amt number); Table created.

SQL> select \* from cust\_mstr; CUST\_NO FNAME LNAME

1 Harshad Karale

2 Aviraj Kale

3 Nayan Keote

SQL> select \* from fd\_dets; FD\_SR\_NO AMT

1 2000

2 6000

3 8000

4 10000

SQL> CREATE TABLE acc\_fd\_cust\_dets(codeno number references cust\_mstr(cust\_no) on delete cascade,acc\_fd\_no number references fd\_dets(fd\_sr\_no) on delete cascade);

Table created.

SQL> select \* from acc\_fd\_cust\_dets; CODENO ACC\_FD\_NO

1 1

2 2

3 3

SQL> select \* from cust\_mstr where cust\_no IN(SELECT codeno from acc\_fd\_cust\_dets where acc\_fd\_no IN(SELECT fd\_sr\_no FROM fd\_dets where amt>5000));

# CUST\_NO FNAME LNAME

2 Aviraj Kale

3 Nayan Keote

* 1. **Create view on cust\_mstr and acc\_fd\_cust\_dets tables by selecting any one column from each table perform insert update delete operations**
     1. Create a view selecting any one column from each table:

CREATE VIEW cust\_acc\_view AS SELECT c.cust\_name, a.acc\_no FROM cust\_mstr c JOIN acc\_fd\_cust\_dets a ON c.cust\_no = a.cust\_no;

* + 1. Perform insert operation on the view:

INSERT INTO cust\_acc\_view (cust\_name, acc\_no) VALUES ('John Doe', 123456);

* + 1. Perform update operation on the view:

UPDATE cust\_acc\_view SET acc\_no = 654321 WHERE cust\_name = 'John Doe';

* + 1. Perform delete operation on the view:

DELETE FROM cust\_acc\_view WHERE cust\_name = 'John Doe';

* 1. **Create following Tables with suitable data and solve following query emp\_mstr(emp\_no,f\_name,l\_name,m\_name,dept)**

**cntc\_dets(code\_no,cntc\_type,cntc\_data)**

**List the employee details along with contact details using left outer join & right join**

SQL> CREATE TABLE branch\_mstr(name char(10),b\_no number primary key);

Table created.

SQL> select \* from branch\_mstr; NAME B\_NO

Computer 1

# MECH 2

IT 3

Civil 4

SQL> CREATE TABLE emp\_mstr(emp\_no number NOT NULL,fname char(10),lname char(10),mname char(10),dept char(10),desg char(10),branch\_no number references branch\_mstr(b\_no) on delete set null);

Table created.

SQL> select \* from emp\_mstr;

# EMP\_NO FNAME LNAME MNAME DEPT DESG BRANCH\_NO

1 Harshad Karale Sanjay Comp Student 1

SQL> select \* from emp\_mstr where branch\_no IN(select b\_no from branch\_mstr where name = 'Computer');

# EMP\_NO FNAME LNAME MNAME DEPT DESG BRANCH\_NO

1 Harshad Karale Sanjay Comp Student 1

1. Create following Tables emp\_mstr(emp\_no,f\_name,l\_name,m\_name,dept) cntc\_dets(code\_no,cntc\_type,cntc\_data) List the employee details along with contact details using left outer join & right join

SQL> CREATE TABLE emp\_mstr(emp\_no number primary key,fname char(10),lname char(10),mname char(10),dept char(10));

Table created.

SQL> select \* from emp\_mstr;

# EMP\_NO FNAME LNAME MNAME DEPT

1. Harshad Karale Sanjay Comp
2. Aviraj Kale Popat IT
3. Nayan Keote Gajanan AIML
4. Ashirwad Katakamwar Rajeshwar Entc
5. Rushi Magadum Ranjit Comp

SQL> CREATE TABLE cntc\_dets(codeno number references

emp\_mstr(emp\_no),cntc\_type varchar(20),cntc\_data varchar(20)); Table created.

SQL> select \* from cntc\_dets; CODENO CNTC\_TYPE CNTC\_DATA

1. Email [harshad@gmail.com](mailto:harshad@gmail.com)
2. Mob 9322918702
3. Email [nayan@gmail.com](mailto:nayan@gmail.com)

3 mob 1234567890 LEFT OUTER JOIN:

SQL> select \* from emp\_mstr LEFT OUTER JOIN cntc\_dets ON emp\_mstr.emp\_no = cntc\_dets.codeno;

# EMP\_NO FNAME LNAME MNAME DEPT CODENO

1. Harshad Karale Sanjay Comp 1
2. Aviraj Kale Popat IT 2
3. Nayan Keote Gajanan AIML 3
4. Ashirwad Katakamwar Rajeshwar Entc
5. Rushi Magadum Ranjit Comp CNTC\_TYPE CNTC\_DATA

Email [harshad@gmail.com](mailto:harshad@gmail.com) Mob 9322918702

Email [nayan@gmail.com](mailto:nayan@gmail.com) RIGHT OUTER JOIN

SQL> select \* from emp\_mstr RIGHT OUTER JOIN cntc\_dets ON emp\_mstr.emp\_no = cntc\_dets.codeno;

# EMP\_NO FNAME LNAME MNAME DEPT CODENO

CNTC\_TYPE CNTC\_DATA

1 Harshad Karale Sanjay Comp 1 Email [harshad@gmail.com](mailto:harshad@gmail.com)

2 Aviraj Kale Popat IT 2

Mob 9322918702

3 Nayan Keote Gajanan AIML 3 Email [nayan@gmail.com](mailto:nayan@gmail.com)

3 Nayan Keote Gajanan AIML 3

mob 1234567890

5. Create following Tables cust\_mstr(cust\_no,fname,lname) add\_dets(code\_no,pincode) List the customer who do not have bank branches in their vicinity.

SQL> CREATE TABLE cust\_mstr(cust\_no varchar(10) primary key ,fname char(10),lname char(10));

Table created.

SQL> select \* from cust\_mstr; CUST\_NO FNAME LNAME

C1 Harshad Karale C2 Aviraj Kale

C3 Nayan Keote

SQL> select \* from add\_dets;

SQL> CREATE TABLE add\_dets(codeno varchar(10),pincode number); Table created.

CODENO PINCODE

B1 410510

C1 410510

C2 410511

C3 410512

SQL> select \* from cust\_mstr where cust\_no IN(select codeno from add\_dets where codeno like 'C%' AND pincode NOT IN(select pincode from add\_dets where codeno like 'B%'));

# CUST\_NO FNAME LNAME

C2 Aviraj Kale C3 Nayan Keote

**Problem statement 5.**

1. **Consider following database schema and solve given queries**

**cust\_mstr(cust\_no,fname,lname)**

**add\_dets(code\_no,add1,add2,state,city,pincode)**

1. **Create above Tables with suitable data**
2. **Retrieve the address of customer Fname as 'xyz' and Lname as 'pqr'**
3. **Create View on add\_dets table by selecting any two columns and perform insert update delete operations**
4. **Create following Tables**

**cust\_mstr(cust\_no,fname,lname) add\_dets(code\_no,pincode)**

(Most of the queries are similar as follow)

**List the customer who do not have bank branches in their vicinity.**

**Problem statement 6.**

**Q 1.Consider table Stud(Roll, Att,Status)**

**Write a PL/SQL block for following requirement and handle the exceptions.**

**Roll no. of student will be entered by user. Attendance of roll no. entered by user will be checked in**

**Stud table. If attendance is less than 75% then display the message “Term not granted” and set the**

**status in stud table as “D”. Otherwise display message “Term granted” and set the status in stud**

**table as “ND”.**

SQL> CREATE TABLE stud(roll\_no NUMBER PRIMARY KEY , att NUMBER ,status char(5)); Table created.

SQL> select \* from stud; ROLL\_NO ATT STATU

123 95 NULL

129 99 NULL

133 94 NULL

137 97 NULL

300 60 NULL

301 70 NULL

Declare

mroll number(10); matt number(10); Begin mroll:=&mroll;

select att into matt from stud where roll\_no = mroll; if matt<75 then

dbms\_output.put\_line(mroll || 'is detained'); update stud set status='D' where roll\_no = mroll; else

dbms\_output.put\_line(mroll || 'is not detained'); update stud set status='ND' where roll\_no = mroll; end if;

Exception

when no\_data\_found then dbms\_output.put\_line(mroll || 'Not found');

End;

/

SQL> select \* from stud; ROLL\_NO ATT STATU

123 95 ND

129 99 ND

133 94 ND

137 97 ND

300 60 D

301 70 D

1. rows selected.

**Q 2.The bank manager has decided to activate all those accounts which were previously marked as**

**inactive for performing no transaction in last 365 days. Write a PL/SQ block (using implicit cursor)**

**to update the status of account, display an approximate message based on the no. of rows affected**

**by the update. (Use of %FOUND, %NOTFOUND, %ROWCOUNT)**

SELECT \* FROM account; ACC\_NO NAME STATUS

129 Harshad active

123 Avi inactive

122 Aadi inactive

137 Nayan active

4 rows selected. BEGIN

UPDATE account SET status = 'active' WHERE status = 'inactive'; dbms\_output.put\_line(SQL%ROWCOUNT||' no of account updated'); END;

/

2 no of account updated

PL/SQL procedure successfully completed. SELECT \* FROM account;

# ACC\_NO NAME STATUS

129 Harshad active

123 Avi active

122 Aadi active

137 Nayan active

4 rows selected.

# BEGIN

UPDATE account SET status = 'active' WHERE status = 'inactive'; dbms\_output.put\_line(SQL%ROWCOUNT||' no of account updated'); END;

/

0 no of account updated

**Problem statement 7.**

**Q 1. Write an SQL code block these raise a user defined exception where business rule is voilated.**

**BR for client\_master table specifies when the value of bal\_due field is less than 0 handle the**

**exception.**

Declare

input number(10); client\_id number; bal\_due Exception;

Begin

client\_id :=& client\_id;

input :=&input; IF input < 0 THEN raise bal\_due; ELSE

INSERT INTO client\_master VALUES(client\_id,input); dbms\_output.put\_line('Inserted successfully');

# END IF;

Exception

when bal\_due then

dbms\_output.put\_line(input || 'Your balance is less then 0'); End;

Enter value for client\_id: 3 old 6: client\_id :=& client\_id; new 6: client\_id :=3;

Enter value for input: -10 old 7: input :=&input;

new 7: input :=-10;

-10

Your balance is less then 0

Enter value for client\_id: 2 old 6: client\_id :=& client\_id;

new 6: client\_id :=2;

Enter value for input: 100 old 7: input :=&input;

new 7: input :=100; Inserted successfully

**Q 2. Organization has decided to increase the salary of employees by 10% of existing salary, who**

**are having salary less than average salary of organization, Whenever such salary updates takes**

**place, a record for the same is maintained in the increment\_salary table. EMP (E\_no , Salary)**

**increment\_salary(E\_no , Salary)**

SELECT \* FROM salary; EMP\_NO SALARY

129 11000

123 22000

137 30000

# DECLARE

CURSOR salhigh IS SELECT emp\_no,salary FROM salary WHERE salary < (SELECT AVG(salary) FROM salary);

memp\_no salary.emp\_no%type; msalary salary.salary%type; BEGIN

OPEN salhigh;

IF salhigh%isopen THEN LOOP

fetch salhigh into memp\_no,msalary; exit when salhigh%notfound;

if salhigh%found then

update salary set salary = (0.1\*msalary+ msalary) WHERE emp\_no = memp\_no; insert into increment\_salary values(memp\_no,0.1\*msalary+ msalary);

end if; end loop; end if;

Close salhigh; END;

/

PL/SQL procedure successfully completed.

SELECT \* FROM salary; EMP\_NO SALARY

129 12100

123 24200

137 33000

SELECT \* FROM increment\_salary; EMP\_NO INCREMENT\_SALARY

121 12100

122 24200

129 33000

**Problem statement 8.**

**Q 1.Borrower(Roll\_no, Name, DateofIssue, NameofBook, Status) Fine(Roll\_no,Date,Amt)**

1. **Accept roll\_no& name of book from user.**
2. **Check the number of days (from date of issue), if days are between 15 to 30 then fine amount**

**will be Rs 5per day.**

1. **If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day. After submitting the book, status will change from I to R**
2. **If condition of fine is true, then details will be stored into fine table.**
3. **Also handles the exception by named exception handler or user define exception handler.**

CREATE PROCEDURE process\_borrower\_fine ( mroll\_no IN NUMBER

# ) IS

days INTEGER; doi DATE;

dor DATE := SYSDATE;

mamt NUMBER;

# BEGIN

SELECT dateofissue INTO doi FROM borrower WHERE roll\_no = mroll\_no; days := dor - doi;

dbms\_output.put\_line(days);

UPDATE borrower SET status = 'r' WHERE roll\_no = mroll\_no;

IF (days > 30) THEN

mamt := (days - 30) \* 50 + 75;

INSERT INTO fine VALUES (mroll\_no, dor, mamt);

ELSIF (days > 15) THEN

mamt := (days - 15) \* 5;

INSERT INTO fine VALUES (mroll\_no, dor, mamt); END IF;

# END;

SQL> BEGIN

2 process\_borrower\_fine(&mroll\_no);

# 3 END;

4 /

SQL> select \* from fine;

ROLL\_NO DATEOFRET AMT

3 16-FEB-24 40

4 16-FEB-24 150

2 16-FEB-24 5

3 16-FEB-24 40

4 rows selected.

**Problem statement 9.**

**Q 1. Write PL/SQL block using explicit cursor for following requirements:**

**College has decided to mark all those students detained (D) who are having attendance less than**

**75%.**

**Whenever such update takes place, a record for the same is maintained in the D\_Stud table.**

**create table stud21(roll number(4), att number(4), status varchar(1)); create table d\_stud(roll number(4), att number(4));**

SELECT \* FROM stud; ROLL\_NO ATT S

-

129 90

137 74

123 50

# DECLARE

CURSOR check\_status IS SELECT roll\_no,att FROM stud WHERE att < 75 ; mroll\_no stud.roll\_no%type;

matt stud.att%type;

# BEGIN

OPEN check\_status;

IF check\_status%isopen THEN LOOP

FETCH check\_status INTO mroll\_no,matt; exit WHEN check\_status%notfound;

IF check\_status%found THEN

UPDATE stud SET status = 'D' WHERE roll\_no = mroll\_no; INSERT INTO d\_stud VALUES(mroll\_no,matt);

END IF;

END LOOP;

END IF;

CLOSE check\_status; END;

/

PL/SQL procedure successfully completed.

SELECT \* FROM stud;

# ROLL\_NO ATT S

-

129 90

# 137 74 D

123 50 D

SELECT \* FROM d\_stud;

# ROLL\_NO ATT

137 74

123 50

**Problem statement 10.**

**Q 1.Consider table Stud(Roll, Att,Status)**

**Write a PL/SQL block for following requirement and handle the exceptions.**

**Roll no. of student will be entered by user. Attendance of roll no. entered by user will be checked in**

**Stud table. If attendance is less than 75% then display the message “Term not granted” and set the**

**status in stud table as “D”. Otherwise display message “Term granted” and set the status in stud**

**table as “ND”.**

**(Taken From ChatGPT)**

# DECLARE

v\_roll\_number Stud.Roll%TYPE; v\_attendance Stud.Att%TYPE; v\_status Stud.Status%TYPE;

# BEGIN

-- Accepting roll number from user v\_roll\_number := &roll\_number;

-- Retrieving attendance and status for the entered roll number SELECT Att, Status

INTO v\_attendance, v\_status FROM Stud

WHERE Roll = v\_roll\_number;

-- Checking attendance percentage and updating status accordingly IF v\_attendance < 75 THEN

DBMS\_OUTPUT.PUT\_LINE('Term not granted');

UPDATE Stud SET Status = 'D'

WHERE Roll = v\_roll\_number; ELSE

DBMS\_OUTPUT.PUT\_LINE('Term granted'); UPDATE Stud

SET Status = 'ND'

WHERE Roll = v\_roll\_number; END IF;

# EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Roll number not found'); WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM); END;

/

**Q 2.Write a update, delete trigger on clientmstr table. The System should keep track of the records**

**that ARE BEING updated or deleted. The old value of updated or deleted records should be added**

**in audit\_trade table. (separate implementation using both row and statement triggers)**

CREATE OR REPLACE TRIGGER trade\_record AFTER INSERT OR DELETE

ON client

# FOR EACH ROW DECLARE

op VARCHAR(10); BEGIN

IF updating THEN

op:='update'; END IF;

IF deleting THEN op:='Delete'; END IF;

INSERT INTO trade VALUES(:old.c\_id,:old.p\_amt,op); END;

/

Trigger created.

SELECT \* FROM client;

# C\_ID P\_AMT

121 1500

122 1800

129 1500

124 1900

125 3000

SELECT \* FROM trade; no rows selected

CREATE OR REPLACE TRIGGER trade\_record AFTER UPDATE OR DELETE

ON client

# FOR EACH ROW DECLARE

op VARCHAR(10); BEGIN

IF updating THEN

op:='update'; END IF;

IF deleting THEN op:='Delete'; END IF;

INSERT INTO trade VALUES(:old.c\_id,:old.p\_amt,op); END;

/

Trigger created.

SELECT \* FROM client;

# C\_ID P\_AMT

121 1500

122 1800

129 1500

124 1900

125 3000

SELECT \* FROM trade; no rows selected

UPDATE client SET p\_amt = 1000 WHERE p\_amt=1500;

4 rows updated. SELECT \* FROM trade; C\_ID P\_AMT STATUS

121 1500 update

129 1500 update

CREATE OR REPLACE TRIGGER trade\_record AFTER UPDATE OR DELETE

ON client DECLARE

op VARCHAR(10); BEGIN

IF updating THEN op:='update'; END IF;

IF deleting THEN op:='Delete'; END IF;

INSERT INTO trade VALUES(:old.c\_id,:old.p\_amt,op); END;

CREATE TABLE trade\_status(status VARCHAR(10)); Table created.

CREATE OR REPLACE TRIGGER trade\_record AFTER UPDATE OR DELETE

ON client DECLARE

op VARCHAR(10); BEGIN

IF updating THEN op:='update'; END IF;

IF deleting THEN

op:='Delete';

# END IF;

INSERT INTO trade\_status VALUES(op); END;

/

Trigger created.

SELECT \* FROM client;

# C\_ID P\_AMT

121 1000

122 1000

129 1000

124 1000

125 3000

UPDATE client SET p\_amt = 1500 WHERE p\_amt=1000; 4 rows updated.

SELECT \* FROM trade\_status;

# STATUS

Update

**Problem statement 11.**

**Q 1. Write a stored function in PL/SQL for given requirement and use the same in PL/SQL block.**

**Account no. and branch name will be accepted from user. The same will be searched in table**

**acct\_details. If status of account is active then display appropriate message and also store the**

**account details in active\_acc\_details table, otherwise display message on screen “account is**

**inactive”.**

SQL> create table acc\_details(acc\_no NUMBER primary key , b\_name char(10) , status char(2));

Table created.

SQL> SELECT \* FROM acc\_details; ACC\_NO B\_NAME ST

1. Nigdi A
2. Nigdi I
3. Nigdi A
4. Khed I
5. Khed I

CREATE OR REPLACE FUNCTION account\_fun ( macc\_no IN NUMBER,

b\_name IN CHAR -- Removed size specification

) RETURN CHAR -- Removed size specification IS

mst CHAR(2); -- It's okay to specify sizes for internal variables BEGIN

SELECT status INTO mst FROM acc\_details

WHERE acc\_no = macc\_no AND b\_name = b\_name; -- Use the parameter b\_name RETURN mst;

# EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN NULL; -- Return NULL if no data is found WHEN OTHERS THEN

RETURN NULL; -- General error handling END;

Function created.

# DECLARE

macc\_no NUMBER(10); mst CHAR(2);

b\_name CHAR(10); -- Example value for b\_name BEGIN

macc\_no := &macc\_no; b\_name := '&b\_name';

mst := account\_fun(macc\_no, b\_name); -- Corrected function name and added b\_name IF mst = 'A' THEN

DBMS\_OUTPUT.PUT\_LINE('Account is active'); -- Corrected quotes and added missing parenthesis

INSERT INTO active\_acc\_details VALUES(macc\_no,b\_name,mst); ELSE

DBMS\_OUTPUT.PUT\_LINE('Account is Inactive'); -- Corrected quotes and added missing parenthesis

# END IF;

END;

SQL> select \* from active\_acc\_details; ACC\_NO B\_NAME ST

# 1 A

1. 1 A

3 Nigdi A

Enter value for macc\_no: 2

old 6: macc\_no := &macc\_no; new 6: macc\_no := 2;

Enter value for b\_name: Nigdi old 7: b\_name := '&b\_name'; new 7: b\_name := 'Nigdi'; Account is Inactive

**Problem statement 12.**

**Q 1. Write an SQL code block these raise a user defined exception where business rule is voilated.**

**BR for client\_master table specifies when the value of bal\_due field is less than 0 handle the**

**exception.**

Declare

input number(10); client\_id number; bal\_due Exception;

Begin

client\_id :=& client\_id; input :=&input;

IF input < 0 THEN raise bal\_due; ELSE

INSERT INTO client\_master VALUES(client\_id,input); dbms\_output.put\_line('Inserted successfully');

# END IF;

Exception

when bal\_due then

dbms\_output.put\_line(input || 'Your balance is less then 0'); End;

Enter value for client\_id: 3 old 6: client\_id :=& client\_id; new 6: client\_id :=3;

Enter value for input: -10 old 7: input :=&input;

new 7: input :=-10;

-10

Your balance is less then 0

Enter value for client\_id: 2 old 6: client\_id :=& client\_id;

new 6: client\_id :=2;

Enter value for input: 100 old 7: input :=&input;

new 7: input :=100; Inserted successfully

**Q 2.Write a before trigger for Insert, update event considering following requirement: Emp(e\_no, e\_name, salary)**

1. **Trigger action should be initiated when salary is tried to be inserted is less than Rs. 50,000/-**
2. **Trigger action should be initiated when salary is tried to be updated for value less than Rs.**

**50,000/-**

**Action should be rejection of update or Insert operation by displaying appropriate error message.**

**Also the new values expected to be inserted will be stored in new table. Tracking(e\_no, salary).**

SELECT \* FROM employee; no rows selected

CREATE OR REPLACE TRIGGER emp\_sal\_record

# BEFORE UPDATE OR INSERT

1. ON employee

# FOR EACH ROW

1. DECLARE
2. sal NUMBER:=:new.salary;

# BEGIN

1. IF sal < 50000 THEN
2. IF updating THEN
3. raise\_application\_error(-20003,'This update opration violet comapany rule / record not inserted');

# END IF;

1. IF inserting THEN
2. raise\_application\_error(-20003,'This insert opration violet comapany rule / record not inserted');

# END IF;

1. ELSE
2. DBMS\_OUTPUT.PUT\_LINE('Record created successfully');
3. INSERT INTO emp\_sal VALUES(:new.e\_no,:new.salary);

# END IF;

1. END;
2. /

Trigger created.

INSERT INTO employee VALUES(129,'Aviraj',100000);

Record created successfully 1 row created.

INSERT INTO employee VALUES(124,'Ankit',1000); INSERT INTO employee VALUES(124,'Ankit',1000)

UPDATE employee SET salary = 1000 WHERE e\_no =129;

**Problem statement 13.**

**Q 1. . Write a PL/SQL stored Procedure for following requirements and call the procedure in appropriate**

**PL/SQL block.**

**Borrower(Rollin, Name, DateofIssue, NameofBook, Status) Fine(Roll\_no,Date,Amt)**

**Accept roll\_no& name of book from user.**

1. **Check the number of days (from date of issue), if days are between 15 to 30 then fine amount**

**will be Rs 5per day.**

1. **If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.**
2. **After submitting the book, status will change from I to R.**
3. **If condition of fine is true, then details will be stored into fine table.**

CREATE PROCEDURE process\_borrower\_fine ( mroll\_no IN NUMBER

# ) IS

days INTEGER; doi DATE;

dor DATE := SYSDATE;

mamt NUMBER;

# BEGIN

SELECT dateofissue INTO doi FROM borrower WHERE roll\_no = mroll\_no;

days := dor - doi; dbms\_output.put\_line(days);

UPDATE borrower SET status = 'r' WHERE roll\_no = mroll\_no;

IF (days > 30) THEN

mamt := (days - 30) \* 50 + 75;

INSERT INTO fine VALUES (mroll\_no, dor, mamt);

ELSIF (days > 15) THEN

mamt := (days - 15) \* 5;

INSERT INTO fine VALUES (mroll\_no, dor, mamt); END IF;

# END;

SQL> BEGIN

1. process\_borrower\_fine(&mroll\_no);

# END;

1. /

SQL> select \* from fine; ROLL\_NO DATEOFRET AMT

3 16-FEB-24 40

4 16-FEB-24 150

2 16-FEB-24 5

3 16-FEB-24 40

4 rows selected.

**Problem statement 14.**

**Q 1. Write a Stored Procedure namely proc\_Grade for the categorization of student. If marks**

**scored by students in examination is <=1500 and marks>=990 then student will be placed in**

**distinction category if marks scored are between 989 and900 category is first class, if marks 899**

**and 825 category is Higher Second Class.**

**Write a PL/SQL block for using procedure created with above requirement. Stud\_Marks(name, total\_marks)**

**Result(Roll,Name, Class)**

SQL> select \* from stud\_marks; ROLL\_NO MARKS

1 1270

2 870

3 970

CREATE OR REPLACE PROCEDURE proc\_Grade(

p\_name IN VARCHAR2, p\_marks IN NUMBER, p\_roll IN NUMBER)

# IS

v\_class VARCHAR2(100); BEGIN

IF p\_marks <= 1500 AND p\_marks >= 990 THEN v\_class := 'Dist';

ELSIF p\_marks < 990 AND p\_marks >= 900 THEN v\_class := 'First';

ELSIF p\_marks < 900 AND p\_marks >= 825 THEN v\_class := 'Higher';

# ELSE

v\_class := 'Other'; END IF;

INSERT INTO Result VALUES (p\_roll, p\_name, v\_class); END;

# DECLARE

v\_name VARCHAR2(100); v\_total\_marks NUMBER; v\_roll NUMBER;

# BEGIN

v\_roll := &v\_roll;

v\_name := '&v\_name';

select marks INTO v\_total\_marks FROM stud\_marks where roll\_no = v\_roll; proc\_Grade(v\_name, v\_total\_marks, v\_roll);

# END;

SQL> select \* from Result; ROLL\_NO NAME CLASS

1 Harshad Dist

2 Nayan Higher

3 Aviraj First

**Problem statement 15. Create Database PCCOE Create following Collections**

**Teachers(Tname,dno,dname,experience,salary,date\_of\_joining ) Students(Sname,roll\_no,class)**

**Q1. Find the information about all teachers**

**Q2. Find the information about all teachers of computer department**

**Q3. Find the information about all teachers of computer,IT,ande&TC department**

**Q4. Find the information about all teachers of computer,IT,and E&TC department having salary**

**greater than or equl to 10000/-**

**Q5. Find the student information having roll\_no = 2 or Sname=xyz**

**Q6. Update the experience of teacher-praveen to 10years, if the entry is not available in database**

**consider the entry as new entry.**

**Q7. Update the deparment of all the teachers working in IT deprtment to COMP Q8. Find the teachers name and their experience from teachers collection**

**Q9. Using Save() method insert one entry in department collection Q10. Using Save() method change the dept of teacher praveen to IT**

**Q11. Delete all the doccuments from teachers collection having IT dept.**

**Q12. Display with pretty() method, the first 3 doccuments in teachers collection in ascending order.**

**Problem statements 16 Consider the relational database Supplier(Sid,Sname,address)**

**Parts(Pid, Pname, Color) Catalog(sid,pid,cost)**

**Q. Find name of all parts whose color is green.**

SELECT Pname FROM Parts

WHERE Color = 'green';

**Q. Find names of suppliers who supply some red parts.**

SELECT DISTINCT Sname

FROM Supplier WHERE Sid IN ( SELECT Sid

FROM Catalog WHERE pid IN ( SELECT Pid

FROM Parts

WHERE Color = 'red'

)

);

**Q. Find names of all parts whose cost is more than Rs25.**

SELECT Pname FROM Parts WHERE Pid IN (

SELECT pid

FROM Catalog WHERE cost > 25

);

**Consider the relational database Person(pname,street city)**

**Company(cname,city)**

**Manages(pname,mname)**

**Q. Find the street and city of all employees who work for “Idea”, live in Pune and earn more than 3000.**

SELECT p.street, p.city FROM Person p

JOIN Manages m ON p.pname = m.pname JOIN Company c ON m.mname = c.cname

WHERE c.cname = 'Idea' AND p.city = 'Pune' AND p.pname IN ( SELECT pname

FROM Manages

GROUP BY pname

HAVING MIN(salary) > 3000

);

**Consider the relational database Student(Rollno,name,address) Subject(sub\_code,sub\_name)**

**Marks(Rollno,sub\_code, marks)**

**Q. Find out average marks of each student along with the name of student.**

SELECT s.name, AVG(m.marks) AS average\_marks FROM Student s JOIN Marks m ON s.Rollno

= m.Rollno GROUP BY s.Rollno, s.name;

**Q. Find how many students have failed in the subject “DBMS”**

SELECT COUNT(\*) AS num\_failed\_students FROM Marks WHERE sub\_code =SELECT sub\_code FROM Subject WHERE sub\_name = 'DBMS') AND marks < 40;

**Problem statements 17**

**Write Pl/SQL code block that will accept account number from user , check if the users balance is less than**

**the minimum balance , only deduct Rs.100/- from the balance .**

CREATE TABLE Account (Acc\_no NUMBER PRIMARY KEY, branch\_name VARCHAR2(50), balance NUMBER);

INSERT INTO Account (Acc\_no, branch\_name, balance)VALUES (1, 'Branch1', 1500); INSERT INTO Account (Acc\_no, branch\_name, balance)VALUES (2, 'Branch2', 2000); INSERT INTO Account (Acc\_no, branch\_name, balance)VALUES (3, 'Branch3', 100);

# DECLARE

v\_acc\_no NUMBER; v\_balance NUMBER;

v\_min\_balance NUMBER := 1000; -- Minimum balance required

# BEGIN

-- Accepting account number from user v\_acc\_no := &acc\_no;

-- Retrieving balance for the entered account number SELECT balance INTO v\_balance

FROM Account

WHERE Acc\_no = v\_acc\_no;

-- Checking if balance is less than minimum balance IF v\_balance < v\_min\_balance THEN

UPDATE Account

SET balance = balance - 100 WHERE Acc\_no = v\_acc\_no;

DBMS\_OUTPUT.PUT\_LINE('Deducted Rs.100 from the balance.'); ELSE

DBMS\_OUTPUT.PUT\_LINE('Balance is above minimum balance.'); END IF;

COMMIT; -- Committing the transaction EXCEPTION

# WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Account number not found.');

# WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM); ROLLBACK; -- Rolling back the transaction in case of error

# END;

/

**Problem statements 18**

**Write Pl/SQL code block for inverting number 1234 to 4321.**

# DECLARE

v\_number NUMBER := 1234; -- Number to invert v\_inverted\_number NUMBER := 0;

v\_remainder NUMBER;

# BEGIN

WHILE v\_number > 0 LOOP

v\_remainder := MOD(v\_number, 10);

v\_inverted\_number := v\_inverted\_number \* 10 + v\_remainder; v\_number := (v\_number - v\_remainder) / 10;

# END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Inverted Number: ' || v\_inverted\_number); END;

/

**Problem statements 19**

**The bank manager has decided to mark all those accounts as inactive (I) on which there are no**

**transactions performed in last 365 days. Whenever any such update takes place a record for the**

**same is maintained in the INACT\_MASTER\_TABLE comprising of the account number, the opening date and type of account. Write PL/SQL code block to do the same(cursor for**

**loop)**

(We doesn’t have for loop cursor)

**Problem statements 20**

**Write PL/SQL code block that will merge the data available in the newly created table**

**NEW\_BRANCHES with the data available in the table BRANCH\_MASTER. If the data in the first**

**table already exists in the second table then data should be skipped.(parameterized cursor)**

(We doesn’t have parameterize cursor)

**Problem statements 21**

**Write PL/SQL code block such that depending upon user supplied account number, the customer to**

**whom account belongs , the introducer of that account are inserted into ACCOUNT\_MASTER\_INFO table .If the user enters an account number that is not in the ACCOUNT\_MASTER table, then the PL/SQL block must display appropriate error message(Exception Handling)**

# DECLARE

v\_acc\_no NUMBER := &account\_number; -- User-supplied account number v\_cust\_name VARCHAR2(100);

v\_introducer VARCHAR2(100);

# BEGIN

SELECT cust\_name, introducer INTO v\_cust\_name, v\_introducer FROM ACCOUNT\_MASTER

WHERE acc\_no = v\_acc\_no;

INSERT INTO ACCOUNT\_MASTER\_INFO (acc\_no, cust\_name, introducer) VALUES (v\_acc\_no, v\_cust\_name, v\_introducer);

DBMS\_OUTPUT.PUT\_LINE('Record inserted successfully.'); EXCEPTION

# WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Account number not found in ACCOUNT\_MASTER table.'); WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM); END;

/

**Problem statements 22**

**A stored function is created to perform the ACCOUNT\_NO check operation**

**.F\_checkAccNO() is**

**the name of function which accept a variable ACCOUNT\_NO and returns the value to host environment The value changes from 0(if ACCOUNT\_NO does not exist) to 1(if**

**ACCOUNT\_NO**

**exist) depending on the records retrieved.**

CREATE OR REPLACE FUNCTION F\_checkAccNO (p\_account\_no IN NUMBER) RETURN NUMBER

# IS

v\_count NUMBER;

# BEGIN

-- Check if the account number exists in the table SELECT COUNT(\*)

INTO v\_count

# FROM ACCOUNT\_MASTER

WHERE acc\_no = p\_account\_no;

-- Return 1 if account number exists, otherwise return 0 RETURN CASE WHEN v\_count > 0 THEN 1 ELSE 0 END;

# END;

/

# DECLARE

v\_result NUMBER;

# BEGIN

v\_result := F\_checkAccNO(123456789); -- Replace 123456789 with the account number you want to check

IF v\_result = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('Account number exists.'); ELSE

DBMS\_OUTPUT.PUT\_LINE('Account number does not exist.'); END IF;

# END;

/

**Problem statements 23**

**create a row level trigger for the CUSTOMERS table that would fire for INSERT or UPDATE or DELETE**

**operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old**

**values and new values**

CREATE OR REPLACE TRIGGER customers\_salary\_trigger BEFORE INSERT OR UPDATE OR DELETE ON customers FOR EACH ROW

# DECLARE

v\_old\_salary NUMBER;

v\_new\_salary NUMBER := :NEW.salary; BEGIN

# IF INSERTING OR UPDATING THEN

v\_old\_salary := :OLD.salary;

DBMS\_OUTPUT.PUT\_LINE('Salary difference: ' || (v\_new\_salary - v\_old\_salary)); ELSIF DELETING THEN

DBMS\_OUTPUT.PUT\_LINE('Salary deleted: ' || :OLD.salary); END IF;

# END;

/

**Problem statements 24**

**Write PL/SQL block to update the Customer table and increase the salary of each customer by 500**

**and use the SQL%ROWCOUNTattribute to determine the number of rows affected.**

# DECLARE

v\_num\_rows NUMBER; BEGIN

-- Update the Customer table to increase the salary by 500 UPDATE Customer

SET salary = salary + 500;

-- Get the number of rows affected by the update v\_num\_rows := SQL%ROWCOUNT;

-- Display the number of rows affected

DBMS\_OUTPUT.PUT\_LINE('Number of rows updated: ' || v\_num\_rows); END;

/